

DTIC FILE COPY

2

NPS-54-90-019

NAVAL POSTGRADUATE SCHOOL

Monterey, California

DTIC
ELECTE
JAN 03 1991
S D D



AD-A230 309

VALIDATING SOFTWARE METRICS

Norman F. Schneidewind

September 1990

Approved for public release; distribution unlimited.

Prepared for:

Naval Surface Warfare Center
Dahlgren, VA 22448

95 1 2 014

NAVAL POSTGRADUATE SCHOOL
Monterey, California

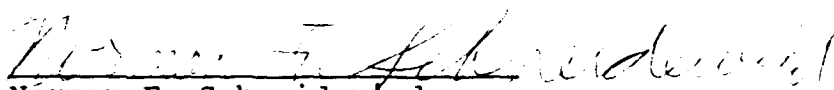
RADM R. W. West, Jr.
Superintendent

Harrison Shull
Provost

The research summarized herein was sponsored by Naval Surface Warfare Center, Dahlgren, Virginia 22448

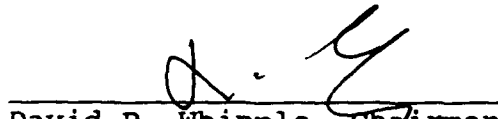
Reproduction of all or part of this report is authorized.

This report was prepared by:



Norman F. Schneidewind
Professor
Department of Administrative
Sciences

Reviewed by:



David R. Whipple, Chairman
Department of Administrative Sciences.

Release by:



Dean of Faculty and Graduate
Studies

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1a REPORT SECURITY CLASSIFICATION Unclassified		1b RESTRICTIVE MARKINGS	
2a SECURITY CLASSIFICATION AUTHORITY		3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; Distribution unlimited.	
2b DECLASSIFICATION/DOWNGRADING SCHEDULE		5 MONITORING ORGANIZATION REPORT NUMBER(S)	
1 PERFORMING ORGANIZATION REPORT NUMBER(S) NPS-AS-90-019		7a NAME OF MONITORING ORGANIZATION	
5a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b OFFICE SYMBOL (if applicable)	7b ADDRESS (City, State, and ZIP Code)	
5c ADDRESS (City, State, and ZIP Code) Monterey, CA 93943		9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N0003090WRO2112AA	
3a NAME OF FUNDING/SPONSORING ORGANIZATION Naval Surface Warfare Center	8b OFFICE SYMBOL (if applicable)	10 SOURCE OF FUNDING NUMBERS	
3c ADDRESS (City, State, and ZIP Code) Dahlgren, VA 22448		PROGRAM ELEMENT NO	PROJECT NO
		TASK NO	WORK UNIT ACCESSION NO
11 TITLE (Include Security Classification) Validating Software Metrics			
12 PERSONAL AUTHOR(S) Norman F. Schneidewind <i>Norman F. Schneidewind</i>			
13a TYPE OF REPORT Technical Report	13b TIME COVERED FROM Dec. 89 TO Sept. 90	14 DATE OF REPORT (Year, Month, Day) 1990 Sept. 30	15 PAGE COUNT 29
16 SUPPLEMENTARY NOTATION			
17 COSATI CODES		18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	Metrics validation methodology; validity criteria; quality functions; non-parametric statistical methods.	
19 ABSTRACT (Continue on reverse if necessary and identify by block number) We propose a comprehensive metrics validation methodology that has six validation criteria, each of which supports certain quality functions. New criteria are defined and illustrated, including consistency, discriminative power, tracking and repeatability. We show that non-parametric statistical methods play an important role in evaluating metrics against the validity criteria. A detailed example of the application of the methodology is presented.			
20 DISTRIBUTION/AVAILABILITY OF ABSTRACT <input type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS		21 ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a NAME OF RESPONSIBLE INDIVIDUAL		22b TELEPHONE (Include Area Code) (408) 646-2719	22c OFFICE SYMBOL

Abstract

We propose a comprehensive metrics validation methodology that has six validation criteria, each of which supports certain quality functions. New criteria are defined and illustrated, including **consistency, discriminative power, tracking and repeatability**. We show that non-parametric statistical methods play an important role in evaluating metrics against the validity criteria. A detailed example of the application of the methodology is presented.

Keywords: metrics validation methodology, validity criteria, quality functions, non-parametric statistical methods.

INTRODUCTION

If the software engineering community believes that the field of metrics should be engineering and not art, then it should subscribe to the idea that we evaluate (validate) whether metrics measure what they purport to measure prior to using the metrics. Furthermore, if metrics are to be of greatest utility, the validation should be performed in terms of the quality functions (quality assessment, control and prediction) that the metrics are to support.

Our purpose is to propose and illustrate a validation methodology whose adoption, we believe, would provide a rational basis for using metrics. We believe this to be the most comprehensive metrics methodology ever proposed. There have been useful validation analyses performed on **specific** metrics or metric systems for the purpose of satisfying **specific** research goals. Among these validations are the following: 1) function points as a predictor of work hours across different development sites and sets of data [2]; 2) reliability of metrics data reported by programmers [3]; 3) Halstead operator count for Pascal programs [7]; 4) metric-based classification trees [18]; 5) evaluation of metrics against syntactic complexity properties [19]. Our approach to validation differs in the following ways: 1) The methodology is **general** and not specific to particular metrics or research objectives. 2) It is developed from the point of view of the **metric user** (rather than the researcher), who has requirements for assessing, controlling and predicting quality. To illustrate the difference in viewpoint, we can make an analogy with the automobile industry: the manufacturer has an interest in brake lining thickness, as it relates to stopping distance, but from the driver's perspective, the only meaningful metric is stopping distance! 3) It consists of six mathematically defined criteria, each of which is keyed to a metrics function, so the user of metrics can understand how a characteristic of a metric, as revealed by validation tests, can be applied to measure software quality. 4) It includes new criteria: **consistency, discriminative power, tracking and repeatability**. 5) It recognizes that a given metric can have **multiple uses** (e.g., **assess, control and predict quality**) and that a given metric can be valid for one use and invalid for another use. 6) It includes some useful statistical methods, rarely seen in the metrics literature, that are applied to metrics validation: **partial linear correlation analysis, chi-square test for differences in probabilities (contingency tables), discriminant analysis and runs test**.

QUALITY
INSPECTED
4

A-1

It is not our purpose to be a proponent or an opponent of given metrics. Whether certain metrics pass or fail our validity tests in the examples is not the point of this paper. The examples are for the sole purpose of illustrating the application of the validation methodology. The validation results could be different in other applications and environments.

We emphasize the use of **non-parametric** statistical techniques for metrics validation because: 1) their application is more consistent with the nature of metrics data (e.g., non-linearity, non-normality, large variability) than are parametric techniques and 2) the measures that result from their application are useful for quality assessment and control.

Outline of Paper

The following subjects are covered:

- o Definitions.
- o Rationale of Metrics Validation.
- o Quality Functions.
- o Non-parametric Statistical Methods for Metrics Validation.
- o Purpose of Metrics Validation.
- o Validity Criteria.
- o Example of Metrics Validation.
- o Summary and Future Research.

DEFINITIONS

Critical Value	Metric value of a validated metric which is used to identify software which has unacceptable quality [11].
Quality Assessment	Evaluation of the relative quality of software components.
Quality Attribute	A feature or characteristic that affects an item's quality [13].
Quality Control	A set of activities designed to evaluate the quality of developed components [modification of 13].
Quality Factor	An attribute of software that contributes to its quality [11]. A quality factor is also a metric.
Quality Metric	A function whose inputs are software data and whose output is a single (numerical) value that can be interpreted as the degree to which software possesses a given attribute that affects its quality [13].
Quality Prediction	A forecast of component quality.

Quality Requirement	A requirement that a software attribute be present in software to satisfy a contract, standard, specification, or other formally imposed document [11].
Software Component	General term used to refer to an element of a software system, such as module, unit, data or document [11].
Software Quality	The degree to which software possesses a desired combination of attributes [12].
Validated Metric	A metric whose values have been statistically associated with corresponding quality factor values [11].

For simplicity of expression, terms will be used without the qualifying word ('metric' instead of 'quality metric') in the remainder of the paper except in the case of 'quality factor' which will be used to distinguish it from 'factor' of the statistical method 'factor analysis'.

RATIONALE FOR METRICS VALIDATION

To help ensure that metrics are used appropriately, only validated metrics (i.e., either quality factors or metrics validated with respect to quality factors) should be used. Quality factors are valid by definition. Furthermore, the metrics which are used should be those which are associated with the quality requirements of the software project. Both product and process metrics are used to assess software quality. Our statements about product elements (i.e., components) apply equally to the processes which produce the products.

It should be understood that if a metric is validated according to our criteria, there is no guarantee that it will faithfully represent a quality factor when applied. Validation is a statistical concept. As such, validation can only be performed within statistical error limits. The major benefit of validation is that it increases the probability that the metric will be a good indicator of quality.

QUALITY FUNCTIONS

Metrics are applied in three major quality functions: **Quality Assessment**, **Quality Control** and **Quality Prediction**. If metrics are to aid in making decisions about software quality, the user of metrics must understand how this tool supports major quality functions in a software engineering organization. Since metrics should not be validated unless the applications of metrics are clearly understood, it is worthwhile to describe the role of metrics during various software phases and the need to validate the metrics for specific metrics functions (i.e., the relationship must be made between (quality functions and validity criteria). Otherwise, a correlation coefficient of .9 between metric X and quality factor Y, for example, is only an abstraction. It only has meaning if validated in the context of quality functions. These purposes are best served by

introducing validity criteria on a qualitative basis now; later, mathematical definitions will be provided in the validation section.

QUALITY ASSESSMENT

Associativity

Software managers need a rational basis for allocating personnel and computer resources to inspection, testing, and other quality activities. A method for doing this is to use metrics to provide a measure of relative quality across components. For example, the magnitudes of a metric are used to establish priority of testing and allocation of budget and effort to testing (i.e., the 'worst' component would receive the most attention, largest budget and most staff). One way to assess relative quality is as follows:

If the elements of a metric vector M , corresponding to components 1, 2, ..., n , are ordered by magnitude, as shown below, does this imply an ordering of component quality?

Magnitude [$M_1 > M_2, \dots, > M_n$] \implies Monotonically Increasing Quality?
(Decreasing)

The validity criterion which assesses the degree to which this relationship is satisfied is called **associativity**. A metric that is validated according to this criterion is used to compare magnitudes of a metric obtained from different components to estimate the degree to which they differ in quality (e.g., 'the quality of Component 2 is twice that of Component 1').

Consistency

It may be that the software manager is only interested in whether 'Component 2 is better than Component 1' rather than how much better. This approach has the advantage of not requiring a linear relationship between quality factors and metrics in order to have perfect association (e.g., if a factor varies as the cube of a metric, there is still perfect association). Thus, rank is the basis of comparison. Therefore, a second way to assess relative quality is as follows:

If the elements of a metric vector M , corresponding to components 1, 2, ..., n , are ordered by rank, as shown below, does this imply an ordering of component quality?

Rank [$M_1 > M_2, \dots, > M_n$] \implies Monotonically Increasing (Decreasing) Quality?

The validity criterion which assesses the degree to which this relationship is satisfied is called **consistency**. A metric that is validated according to this criterion is used to compare ranks of a metric obtained from different components to order the quality of a set of components.

QUALITY CONTROL

Discriminative Power

Metrics are used to monitor the condition of a component to determine whether the component appears to be out of tolerance. This is defined to be a component whose quality is below standard. This implies that critical values of metrics must be established prior to the monitoring activity for comparing against the measured values derived from the component.

In order to control quality during the design phase, components are identified which appear to have unacceptable quality. Unacceptable quality may be manifested as excessive complexity, inadequate documentation, lack of traceability, or other undesirable attributes. The existence of such conditions is an indication that the software may not satisfy quality requirements when it becomes operational. Since many of the quality factors which are usually of interest (e.g., reliability), cannot be measured during design, and are only available during test and operation, validated metrics are used when quality factors are not available. Validated metric measurements are compared with the critical values of the metrics. Components whose measurements are greater than (or less than) the critical values are flagged for detailed inspection. Depending on the results of the inspection, components are redesigned, scrapped, or not changed. The fact that a measurement is outside the critical value does not necessarily mean that the component will exhibit unacceptable quality during operation; rather, it is a warning that the condition bears investigation. This concept is illustrated in Figure 1 for metric vector M for components 1, 2, ..., n . The role of metrics validation for this use of quality control is to identify a critical value of a metric, where that metric has been validated against a quality factor on a previous similar project. Then the metric can serve as a substitute to identify unacceptable quality during design. Such a metric satisfies the discriminative power validity criterion.

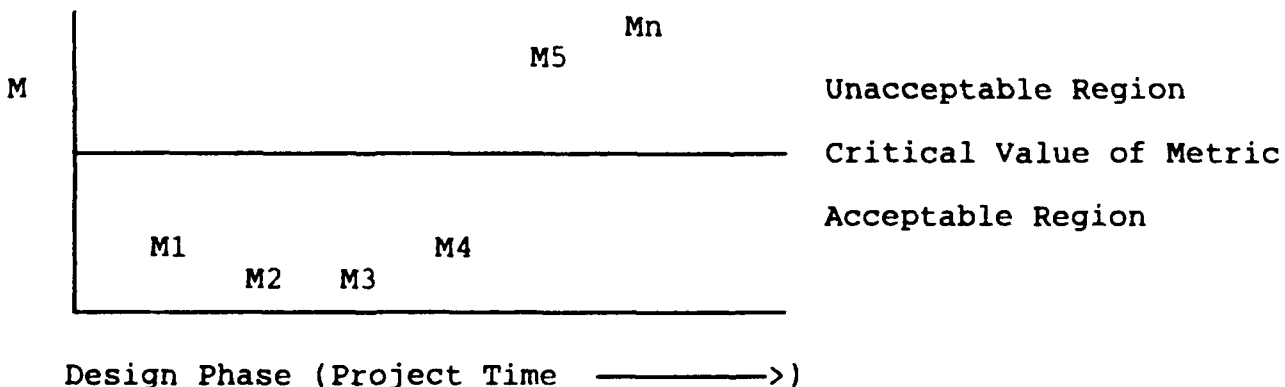


Figure 1. Application of Metrics to Quality Control (discriminative power)

Tracking

In addition to component quality lying within acceptable bounds, a desirable condition is for quality to improve over the life of the component (i.e., a component should exhibit quality growth). Thus, during all phases of the life of the component we wish to track quality in order to control quality. That is, we want to know whether the software is getting better, worse, or staying the same. Again, in most phases, the quality factor will not be available but we must know how quality might be changing, nevertheless. This concept is illustrated in Figure 2 for metric vector M for a given component i , measured at times T_1, T_2, \dots, T_n . In this illustration, quality increases from T_1 to T_2 , stays the same from T_2 to T_3 , and decreases from T_3 to T_n , assuming high metric values are 'bad'. Here, the question for metrics validation is whether a metric can be identified whose changes over time will track changes in quality. In particular, if a metric has been validated as tracking a quality factor on a previous similar project, it would serve as a substitute for tracking quality on the given project. Such a metric satisfies the tracking validity criterion.

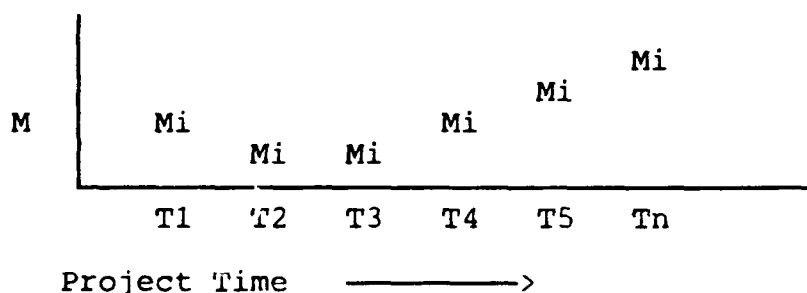


Figure 2. Application of Metrics to Quality Control (tracking)

QUALITY PREDICTION

Predictability

During the design phase validated metrics are used to make predictions of test or operational phase quality factors. Predicted values of quality factors are compared with target values. Components whose predicted quality factor values are greater than (or less than) the target values are flagged for detailed inspection. Potentially, prediction is more valuable than assessment and control because it estimates the attribute of ultimate interest -- the quality factor. However, prediction is more difficult because it involves using validated metrics from an early phase (e.g., design) to make predictions about a different but related attribute (quality factor) in a much later phase (e.g., operations). This concept is illustrated in Figure 3 where, at time T_1 , metric M is used to predict the factor F_p at time T_2 , for a given component, and F_a is eventually observed as the actual value at T_2 . The challenge to metrics validation is to find a metric or metrics that can predict a quality factor with acceptable accuracy. Such a metric satisfies the predictability validity criterion.



Figure 3. Application of Metrics to Quality Prediction(predictability)

NON-PARAMETRIC STATISTICAL METHODS FOR METRICS VALIDATION

Among the advantages of non-parametric statistical methods over parametric methods [5,6,8] which are important for metrics validation, are the following:

- o Assumptions less restrictive than with parametric methods. Given the noisiness of metrics data, this is a big plus.
- o No assumption about distribution (e.g., data does not have to be normally distributed).
- o Can use ordinal scale (i.e., component A is higher quality than component B).
- o Can use nominal scale (i.e., A is high quality; B is low quality)
- o Do not need interval scale (i.e. difference between A quality and B quality).
- o Do not need ratio scale (i.e., A is 2.5 the quality of B).

For example, ranks of random variables [3] can be used rather than the values themselves, thus relaxing the assumptions about data relationships (e.g., linearity) while providing a measure of quality (e.g., ranking of components) that is useful to the software manager. In other words the fact that the data is not as 'well-behaved' as we might believe it should be does not necessarily mean that it is less useful. In fact, when we consider that many useful applications of metrics can be derived from the ability to **classify components** as being 'better' or 'worse', 'high quality' or 'low quality', acceptable or unacceptable, we realize that the information provided by non-parametric analysis is supportive of this approach..

Multivariate statistical methods (e.g., correlation analysis, factor analysis) are also used where appropriate.

PURPOSE OF METRICS VALIDATION

The purpose of metrics validation is to identify metrics that are related to quality factors. If metrics are to be useful, they must indicate accurately whether quality requirements have been achieved or are likely to be achieved in the future. When it is possible to measure quality factors at the desired point in the life of the software, they are used to evaluate software quality. At other points, certain quality factors (e.g., reliability) are not available; they are obtained after delivery or late in the project. In these cases, metrics are used early in a project to **assess, control and predict** quality.

It is important that metrics be validated before they are used to evaluate software quality. Otherwise, metrics may be misapplied (i.e., metrics may be used that have little or no relationship to the desired quality characteristics).

VALIDITY CRITERIA

To be considered valid, a metric must demonstrate a high degree of association with the quality factor it represents. A metric may be valid with respect to certain validity criteria and invalid with respect to other criteria.

The validation procedure requires that threshold values of validity criteria be selected. These are the values 'V', 'B', 'A', and 'P' which are described below. The criterion used for selecting these values is reasonableness (i.e., judgement must be exercised in selecting values to strike a balance between the one extreme of causing a metric which has a high degree of association with a quality factor to fail validation and the other extreme of allowing a metric of questionable validity to pass validation).

A short numerical example follows the definition of each validity criterion.

Note: As previously stated, there are many advantages to using the **general** class of non-parametric statistical methods for metrics validation. However, although the **specific** methods that are associated with each validity criterion are appropriate, they are not necessarily the only methods that could be used.

Associativity: The variation in the quality factor explained by the variation in the metric, which is given by the square of the linear correlation coefficient (R) between the metric and the corresponding quality factor, must exceed V ($R^2 > V$).

This criterion assesses whether there is a sufficiently strong linear association between a quality factor and a metric to warrant using the metric as a substitute for the quality factor, when it is infeasible to use the latter. This criterion supports the quality assessment function. The multivariate statistical methods of linear correlation and partial linear correlation analysis [15] can be used for this test.

For example, the correlation coefficient between a complexity metric and the quality factor reliability may be .9. The square of this is .81. Thus 81% of the variation in the quality factor is explained by the variation in the metric. If this relationship is demonstrated over a representative sample of components, and if V has been established as .7, one could conclude that the metric has the ability to associate complexity with reliability and can be used to compare magnitudes of complexity obtained from different components to estimate the degree to which they differ in reliability.

Consistency: If a quality factor vector F_1, F_2, \dots, F_n , corresponding to components 1, 2, ..., n, has the relationship $F_1 > F_2 > \dots, F_n$, the corresponding metric vector must have the relationship $M_1 > M_2 > \dots, M_n$.

This criterion assesses whether there is consistency between the ranks of the quality factor and the ranks of the metric for the same set of components. Thus this criterion is used to determine whether a metric can accurately rank, by quality, a set of components. This criterion supports the quality assessment function. The non-parametric statistical method Spearman Rank Correlation [3,5,6,8] can be used for this test.

For example, if the reliability of components A, B and C, as measured by MTTF, is 1000, 1500 and 800 hours, respectively, and the corresponding complexity metric values are 5, 3 and 7, where low metric values are 'better' than high values, the ranks for reliability and metric values, with '1' representing the 'highest' rank, are as follows:

Component	Reliability Rank	Complexity Rank
B	1	1
A	2	2
C	3	3

If this relationship is demonstrated over a representative sample of components, one could conclude that the metric is consistent and can be used to rank the quality of components.

Discriminative Power: A metric must be able to discriminate between high quality components (e.g., high MTTF) and low quality components (e.g., low MTTF). For example, the set of metric values associated with the former should be significantly higher (or lower) than those associated with the latter.

This criterion assesses whether a metric is capable of separating a set of high quality components from a set of low quality components. This capability allows one to establish critical values for metrics which can be used to identify components which may have unacceptable quality. This criterion supports the quality control function. The following non-parametric statistical methods can be used for this validation test: Mann-Whitney Test [4,5,6,8], chi-square test for differences in probabilities (contingency tables) [5,8] and the Krusal-Wallis Test [4,5,6,8]. The multivariate statistical method discriminant analysis [1,15] can also be used.

For example, if all components with a complexity metric value of >10 (critical value) have a MTTF of 1000 hours and all components with a complexity metric value equal to or less than 10 have a MTTF of 2000 hours, and this difference is sufficient to pass the statistical tests, then the metric separates low from high quality components. If the ability to discriminate is demonstrated over a representative sample of software components, one could conclude that the metric can discriminate between low and high reliability components.

Tracking: If a metric M is directly related to a quality factor F , for a given component, then a change in a quality factor value from F_{T1} to F_{T2} , at times $T1$ and $T2$, must be accompanied by a change in metric value from M_{T1} to M_{T2} , which is the same direction (e.g., if F increases, M increases). If M is inversely related to F , then a change in F must be accompanied by a change in M in the opposite direction (e.g., if F increases, M decreases).

This criterion assesses whether a metric is capable of tracking changes in quality over the life of a component. This criterion supports the quality control function. The following non-parametric statistical methods can be used for this validation test: Spearman Rank Correlation and Wald-Wolfowitz Runs Test (test for randomness) [5,8].

For example, if a complexity metric is claimed to be a measure of reliability, then it is reasonable to expect a change in the reliability of a component to be accompanied by an appropriate change in metric value (e.g., if the component increases in reliability, the metric value should also change in a direction that indicates the component has improved). That is, if MTTF is used to measure reliability and is equal to 1000 hours during testing($T1$) and 1500 hours during operation ($T2$), a complexity metric whose value is 8 in $T1$ and 6 in $T2$, where 6 is 'better' than 8 (i.e., complexity has decreased), is said to track reliability for this component. If this relationship is demonstrated over a representative sample of components, one could conclude that the metric can track reliability (i.e., indicate changes in component reliability) over the life of the component.

Predictability: If a metric is used at time $T1$ to predict a quality factor for a given component, it must predict a related quality factor Fp_{T2} with an accuracy of:

$$\left| \frac{Fa_{T2} - Fp_{T2}}{Fa_{T2}} \right| < A$$

where Fa_{T2} is the actual value of F at time $T2$.

This criterion assesses whether a metric is capable of predicting a quality factor value with the required accuracy. It is simply a relative error calculation [2,6], that takes into consideration the time of measurement. The multivariate statistical methods of linear regression, multiple linear regression, and non-linear regression can be used for this analysis.

For example, if a complexity metric is used during design to predict the reliability of a component during operation (T_2) to be 1200 hours MTTF (Fp_{T_2}) and the actual MTTF that is measured during operation is 1000 hours (Fa_{T_2}), then the error in prediction is 200 hours, or 20%. If the acceptable prediction error (A) is 25%, prediction accuracy is acceptable. If the ability to predict is demonstrated over a representative sample of components, one could conclude that the metric can be used as a predictor of reliability. For example, prediction could be used during design to identify those components that need to be improved.

Repeatability: A metric must demonstrate the above associativity, consistency, discriminative power, tracking, and predictability properties for P percent of the applications of the metric.

This criterion is used to ensure that a metric has passed a validity test over a sufficient number or percentage of applications so that there will be confidence that the metric can perform its intended function consistently.

For example, if the required 'success rate' (P) for validating a complexity metric against the Predictability criterion has been established as 80%, and there are 100 components, the metric must predict the quality factor with the required accuracy for at least 80 of the components.

VALIDATION PROCEDURE

Metrics validation includes the following steps:

o Identify the Quality Factors Sample

Draw a random sample from the metrics database.

o Identify the Metrics Sample

Draw a random sample from the same domain (e.g., same software) of the metrics data base.

o Perform Goodness of Fit Tests

Perform goodness of fit tests on the quality factor and metrics data to identify their distributions.

o Perform a Statistical Analysis

Perform a statistical analysis using the methods listed under Validity Criteria.

o Re-validate Metrics

Metrics validation is a continuous process. It is important to revalidate a metric each time it is used. As the software engineering process changes, the validity of metrics changes. A validated metric may not necessarily be valid in other environments or future applications. A metric that has been invalidated may be valid in other environments or future applications.

o Validate and Apply Metrics in Similar Environments

There have been great disparities in results reported in the literature concerning 'relationships' between metrics and the quantities they purport to measure. For example, correlation coefficients of number of errors with Halstead Effort and McCabe Complexity differ by a factor of almost two [11]. Differences have also been reported with respect to specification refinement levels [10]. These disparities point up the need to apply metrics under conditions that are similar to those used to validate the metrics.

There should be a project in which metrics data have been collected and validated prior to application of the metrics. This project should be similar to the one in which the metrics are applied, with respect to application, project size, software engineering environment, design methodology, and programming language. In other words, to the extent possible, conduct a **controlled experiment** [6]. Validation and application of metrics should be performed during the same phases on different projects. Example: if metric X is collected during the design phase of project A and the saved values are later validated with respect to quality factor Y, which is collected during the operations phase of project A, the metric X should be used during the design phase of project B to assess quality factor Y with respect to the operations phase of project B.

EXAMPLE OF VALIDATING METRICS

The following example is provided to show how to make metric validation tests. No inferences should be drawn from this example regarding the validity of these metrics for other applications. These metrics are used for illustrative purposes only. The results of the validation tests could be different for other applications. The data used in the validation tests were collected from actual software projects.

Purpose of Metrics Validation

The purpose of this validation is to determine whether cyclomatic number (**complexity (C)**) and size (**number of source statements (S)**) metrics, either singly or in combination, could be used to **assess, control and predict** the quality factor reliability, as represented by the quality factor error count (E).

Validity Criteria

Select values of V, B, A, and P. The values of V, B, A, and P, used in the example are .7, .7, 20%, and 80%, respectively.

VALIDATION PROCEDURE

Perform the following validation steps:

Identify the Quality Factor Sample

Draw a random sample of procedures (i.e., components), which is summarized in Table 1, from the metrics data base, for the quality factor reliability, which is represented by the quality factor error count (Errors). The error counts are listed by project and procedure in Appendix A.

Identify the Metrics Sample

Using the same procedures (i.e., components) in Table 1, identify the metrics samples for cyclomatic number (**complexity**) and size (**statements**). The metrics values are listed by project and procedure in Appendix A.

Table 1

Project Application		Procedures (with errors)	Statements	Errors
1	String Processing	11 (5)	136	10
2	Directed Graph Analysis	31 (12)	430	27
3	Directed Graph Analysis	1 (1)	13	1
4	Data Base Management	69 (13)	1021	26
		112 (31)	1600	64

Number of procedures: 112 total, 31 with errors, 81 with no errors.

Number of source statements: 2007 total, 1600 included in metrics analysis.

Language : Pascal on all projects.

Programmer: Single programmer. Same programmer on all projects.

Perform Goodness of Fit Tests

The best fits obtained for the data are the following distributions:

Errors: Negative Binomial (error procedures)

Complexity: Negative Binomial (all procedures)

Statements: Exponential (all procedures)

Thus, this result discourages the use of statistical methods that depend on assumptions of normality and encourages the use of non-parametric methods.

Perform a Statistical Analysis

Perform the tests described under Validity Criteria. Significance level and sample size are denoted by α and N , respectively; when it is necessary to specify a critical level of α in hypothesis tests, .05 is used.

Associativity

1. Compute the sample linear correlation coefficient (R) for Errors (E) and Complexity (C) and for Errors (E) and Statements (S) and compare each R^2 with $V = .7$ [15].

Table 2
Sample Correlations (Error Procedures)
N = 31

	Complexity	Statements
Errors	.7834	.5880
α	.0000	.0005

Sample Correlations (All Procedures)
N = 112

	Complexity	Statements
Errors	.8010	.6596
α	.0000	.0000

RESULT: $R^2 < V = .7$. Fails minimum R^2 tests.

2. Perform a null hypothesis test $H_0: \rho = 0$ for E and C. [15].

RESULT: Reject H_0 with $\alpha = .0000$ and $N = 31$.

3. Perform a null hypothesis test $H_0: \rho > \sqrt{V} = .836$ for E and C, since we want $R^2 > V = .7$ [15].

RESULT: Accept H_0 with $\alpha = .01$ and $N = 31$.

4. Compute the partial correlation coefficients for E, C, and S. These coefficients give the strength of the linear relationship between two variables while controlling for the effects of the remaining variables [15]. This is a method for controlling for the effect of size (i.e., when the partial correlation coefficient between E and C is computed, the effect of S is eliminated so that the association between E and C alone can be observed).

Table 3
Sample Partial Correlations (Error Procedures)
N = 31

	Complexity	Statements
Errors	0.64298	-0.08157
Complexity		0.65568

RESULT: From the low R for E and S, it can be seen that Statements contributes essentially no additional information about Errors, once Complexity has been correlated with Errors. Also, the R for E and C indicates the correlation between Errors and Complexity with the effect of size (S) eliminated.

5. Compute a confidence interval of p for E and C [15].

RESULT: $.593 < p < .891$ with $\alpha = .05$ and $N = 31$.

Tests 3, 4 and 5 provide additional useful information about linear correlation but they are not part of the required validation procedure.

6. Perform a Factor Analysis

Note: In this section a factor is defined as follows:

$X_j = \lambda_{1j}F_1 + \lambda_{2j}F_2 + \dots + \lambda_{kj}F_k + U_j$, where

X_j is a variable (metric), F_1, F_2, \dots, F_k are factors that are common to all the variables, U_j is a random factor unique to X_j ,

and $\lambda_{1j}, \lambda_{2j}, \dots, \lambda_{kj}$ are factor loadings

(correlations between variables and factors) [9,15].

Do not confuse the use of the statistical term 'factor' with the use of the metrics term 'quality factor'.

The objective of factor analysis is to reduce a set of metrics to a smaller, orthogonal set of factors that can better explain the relationships between correlated metrics. It frequently occurs that several 'independent' variables (Complexity, Statements) that are used to study the behavior of a dependent variable (Errors) are themselves dependent and correlated -- the multicollinearity problem (See Table 3). Recent studies [14,17] have shown that a large number of metrics [16] can be reduced to a small manageable set that represents the underlying relationship between the **quality factor** and one or more metrics. The method is most useful when there are many metrics. The example that follows only involves three metrics. The mechanics of the analysis are to attempt to identify one or more factors that contain high loadings for a subset of the metrics in the factor, including the **quality factor**, and low loadings for the remaining metrics. Then the loadings are examined, excluding the **quality factor**, to see which metrics of the candidate factors from the first step have high loadings. These metrics would be emphasized in certain other analyses, like regression analysis. The remaining metrics would be deemphasized or discarded. An example is shown in Table 4, for procedures with errors, where Factor 2 contains relatively high loadings for 'Errors' and 'Complexity'. Table 5 shows a relatively high loading for 'Complexity'. This analysis indicates that a single metric -- Complexity -- suffices for explaining the variance in the Errors metric. A similar result was obtained using all procedures.

Table 4
Factor Loadings (Error Procedures)
N = 31

Metric	Factor 1	Factor 2
Errors	0.30128	0.94013
Complexity	0.68442	0.66467
Statements	0.94057	0.29572

Table 5
Factor Loadings (Error Procedures)
N = 31

Metric	Factor 1	Factor 2
Complexity	0.44002	0.89799
Statements	0.89799	0.44002

CONCLUSION: The results are mixed. Although the results of tests 2, 3 and 5 are favorable, Complexity failed mandatory Test 1. Thus, evaluating the results conservatively, Complexity is judged to be invalid with respect to Associativity. Statements does not perform as well as Complexity and is invalid with respect to Associativity. Furthermore the factor analysis indicates that only one of the metrics -- Complexity -- is needed.

Consistency

1. Compute the Spearman Coefficient of Rank Correlation (r) for E and C over all procedures with errors. Correlation is lower for E and S than for E and C and is not shown. Compare r with $B = .7$ and α with .05 [5,8].

Table 6
Spearman Rank Correlation (Error Procedures)
N = 31

	Complexity	Remarks
Errors	.5119	$r < .7$
α	.0051	$\alpha < .05$

RESULT: The desired result is $r > .7$ and $\alpha < .05$. Complexity does not change consistently with changes in Errors across all procedures with errors. Therefore Complexity is not valid with respect to Consistency. Also, Statements is not valid with respect to Consistency.

Discriminative Power

1. Divide the data into two sets: procedures with errors and procedures with no errors. Rank these sets according to their C and S values (statistical programs will do the ranking automatically) and perform the Mann-Whitney test to see whether C and S can discriminate between the two sets of procedures (i.e., tell the difference between high quality and low quality software) [5,8].

RESULT: The results of the Mann-Whitney test for C and S are shown in Table 7. The average ranks of C and S for procedures with errors are much higher than the average ranks for procedures with no errors, respectively. We can infer from the low probabilities of higher statistics that C and S for procedures with errors have significantly higher medians in the populations (i.e., that C and S could discriminate apriori between high quality and low quality software). Caution: a large number of ties weakens this test. There are a large number of ties in C but not in S [5,8].

Table 7
Mann-Whitney Test: Comparison of Two Samples

Sample 1: Complexity - Procedures with errors

Sample 2: Complexity - Procedures with no errors

Average rank of first group = 85.9032 based on 31 values.

Average rank of second group = 45.2469 based on 81 values.

Large sample test statistic Z = -6.30181

Two-tailed probability of equaling or exceeding Z = 2.95465E-10

N: 112 total observations.

Sample 1: Statements - Procedures with errors

Sample 2: Statements - Procedures with no errors

Average rank of first group = 85.2419 based on 31 values.

Average rank of second group = 45.5 based on 81 values.

Large sample test statistic Z = -5.82408

Two-tailed probability of equaling or exceeding Z = 5.76106E-9

N: 112 total observations.

2a. Divide the data into four categories, as shown in Table 8, according to a critical value of C, C_c , so that a Chi-square test can be performed to determine whether C_c can discriminate between procedures with errors and those with no errors. C_c is chosen to provide at least five observations for each cell in Table 8 in order to ensure the validity of the test. This may involve trial and error [5].

Table 8
Contingency Table

	Complexity ≤ 3	Complexity > 3	
No Errors	75	6	81
Errors	10	21	31
	85	27	112

RESULT: The result of the Chi-square test is shown in Table 9. From the high value of chi-square and the very small significance level in the samples, we infer that C_c could discriminate between procedures with errors (low quality software) and those without errors (high quality software).

Table 9

Summary Statistics for Contingency Tables: $C_c = 3$

Chi-square	D.F.	Significance
44.6081	1	2.40692E-11

Sensitivity Analysis of Critical Value of Complexity

In order to see how good a discriminator C_c is for this example, we observe the number of misclassifications that result for various values of C_c : 1) Type 1 ('error procedures' classified as 'no error procedures') and 2) Type 2 ('no error procedures' classified as 'error procedures'). This is shown in Figure 4. As C_c increases, Type 1 misclassifications increase because an increasing number of high complexity procedures, many of which have errors, are classified as having 'no errors'. Conversely, as C_c decreases, Type 2 misclassifications increase because an increasing number of low complexity procedures, many of which have no errors, are classified as having 'errors'. The total of the two curves represents the 'misclassification' function. It has a minimum at $C_c = 3$, which is the value given by the Chi-square test (the Chi-square test will not always produce the optimum C_c but the value should be close to optimum).

—●— TYPE 1
 -+ - TYPE 2
 ·*· TOTAL

INCORRECT CLASSIFICATION
 (COMPLEXITY)

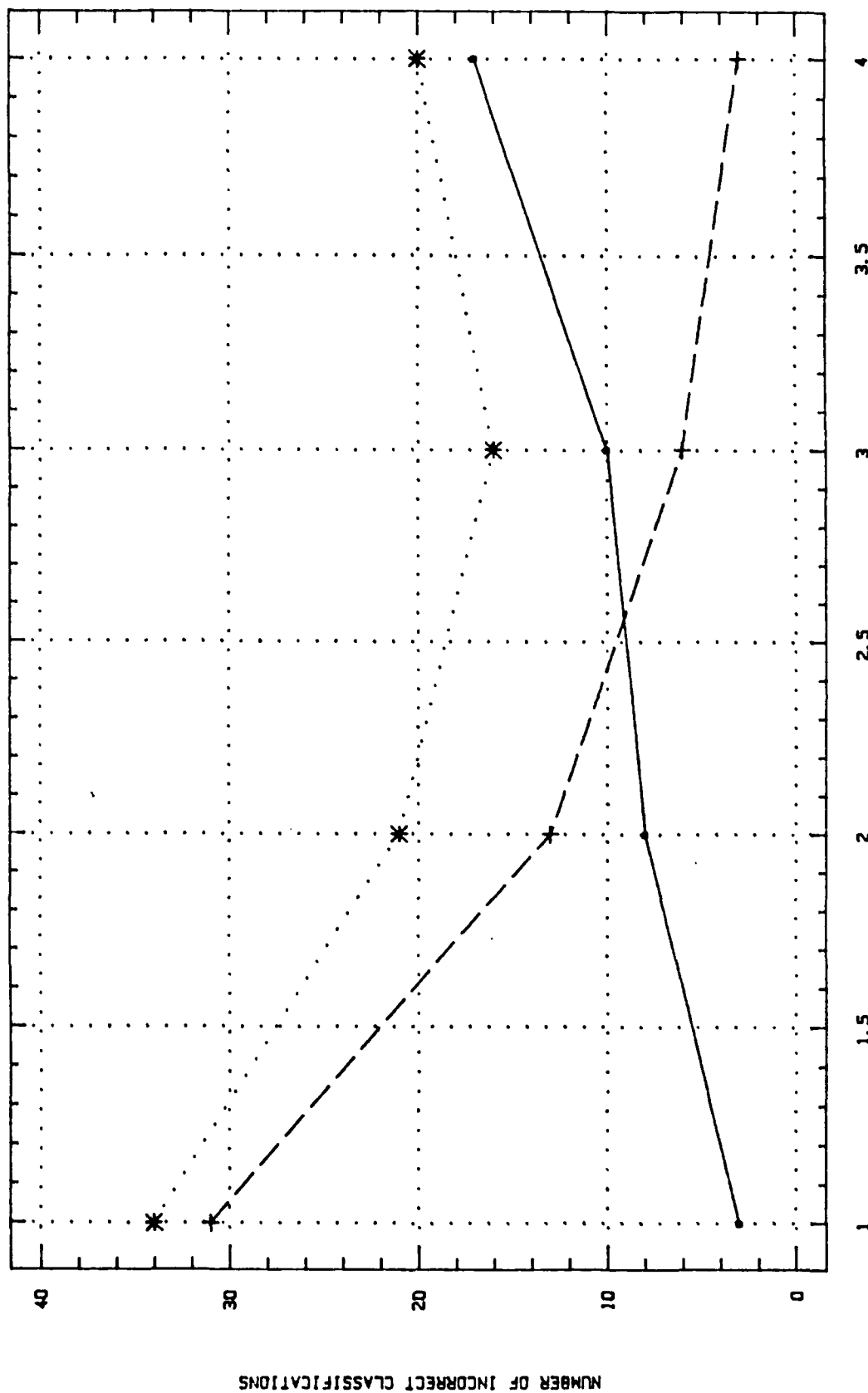


Figure 4

The foregoing analysis assumes that the costs of Type 1 and Type 2 misclassifications are equal. This is usually not the case since the consequences of not finding an error (i.e., concluding that there is no error when, in fact, there is an error) would be higher than the other case (i.e., concluding that there is an error when, in fact, there is no error). In order to account for this situation, the number of Type 1 misclassifications, for given values of C_c , is multiplied by $C1/C2$ ($C1/C2 = 1, 2, 3, 4, 5$), which is the ratio of the cost of Type 1 misclassification to the cost of Type 2 misclassification. These values are added to the number of Type 2 misclassification to produce the family of five 'cost' curves shown in Figure 5. Naturally, with the higher cost of Type 1 misclassifications taking effect, the optimum C_c (i.e., minimum cost) decreases. However, even at $C/C2 = 5$, $C_c = 3$ is a reasonable choice.

2b. Do Step 2a. for S. The Contingency Table is shown in Table 10.

Table 10

Contingency Table

Statements Statements

≤ 13 > 13

No Errors	64	17	81
Errors	7	24	31
	71	41	112

Table 11

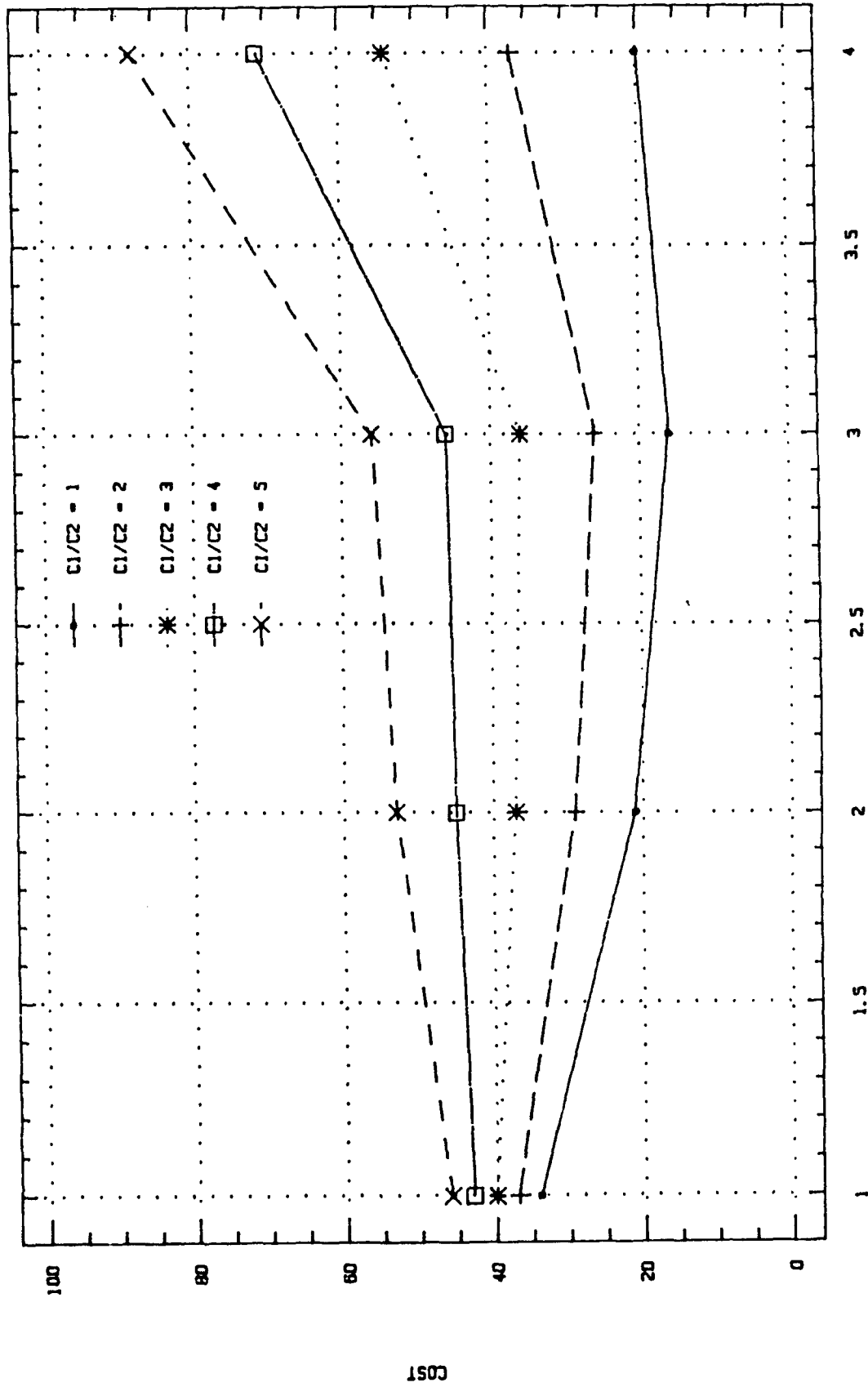
Summary Statistics for Contingency Tables: $S_c = 13$

Chi-square	D.F.	Significance
30.7658	1	2.91118E-8

RESULT: The same comments made in Step 2a. apply to S_c .

COST OF INCORRECT CLASSIFICATION

(COMPLEXITY)



CRITICAL VALUE OF COMPLEXITY

Figure 5

Sensitivity Analysis of Critical Value of Size

The same type of analysis is performed on S_c as was performed on C_c to see how good S_c is as a discriminator of quality. The curves of Type 1, Type 2 and total misclassifications are shown in Figure 6, where it is seen that the optimum $S_c = 15$, as opposed to $c = 13$, as given by the Chi-square analysis. The 'cost' curves are shown in Figure 7, where again the optimal S_c decreases as $C1/C2$ increases. Considering the family of cost curves, $S_c = 13$ is a reasonable value but S_c does not perform as well as C_c in this example, because, whereas $S_c = 15$ is not optimum for any of the cost curves, $C_c = 3$ is optimum for three of the five curves. This result could be anticipated by the higher Chi-square and lower value of significance (better ability to distinguish between high and low quality) obtained for C in Table 9 as compared to the corresponding values obtained for S in Table 11.

3. Perform the Krusal-Wallis test (not shown) to ascertain whether C and S are good discriminators with respect to given values of E (i.e., higher ranks of C and S for higher values of E).

RESULT: C and S were good discriminators when both procedures with errors and all procedures were evaluated.

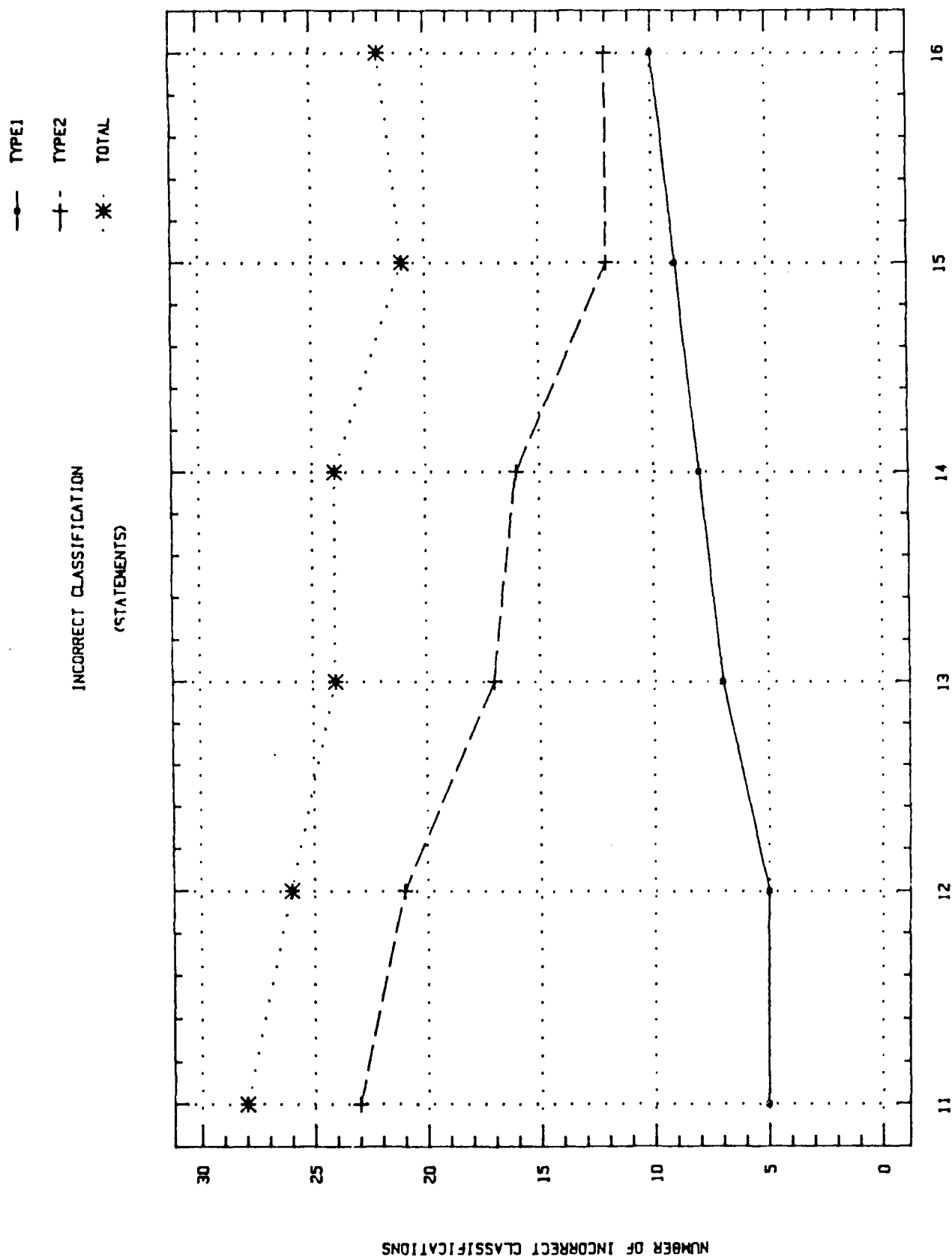
Discriminant Analysis

Another approach to estimating and using a critical value of a metric is to use discriminant analysis [1,15]. We briefly describe this method more to indicate its general potential than as a method that can be applied in this example because, unfortunately, discriminant analysis is based on the assumption that the random variables are normally distributed [1,15]. This is not the case for E , C and S , as was observed from the goodness of fit tests.

In this technique, a linear function of random variables, called the discriminant function, is found such that, when this function is evaluated, its value can be used to classify the random variables into one of N groups. For example, a linear function of C and S :

$$L = b_c C + b_s S$$

can be used to classify the tuple (C,S) into the 'error' group or 'no error' group depending on whether $L \geq$ or $< \bar{L}$ the cutoff value of the discriminant function. The coefficients of L are determined by maximizing the ratio of the variance between the two groups to the variance within groups, thus providing for maximum discrimination. Using \bar{C}_e and \bar{S}_e of the 'error' group, a cutoff value \bar{L}_e is determined for the 'error' group. Similarly, using \bar{C}_{ne} and \bar{S}_{ne} of the 'no error' group, a cutoff value \bar{L}_{ne} is determined for the 'no error' group. The two values of L are combined to form a single cutoff value \bar{L} . A reasonable way to do this is to weight \bar{L}_e by the probability of a component being in the 'error' group (i.e., the fraction of components in the 'error' group) and to weight \bar{L}_{ne} by the probability of a component being in the 'no error' group (i.e., the fraction of components in the 'no error' group).

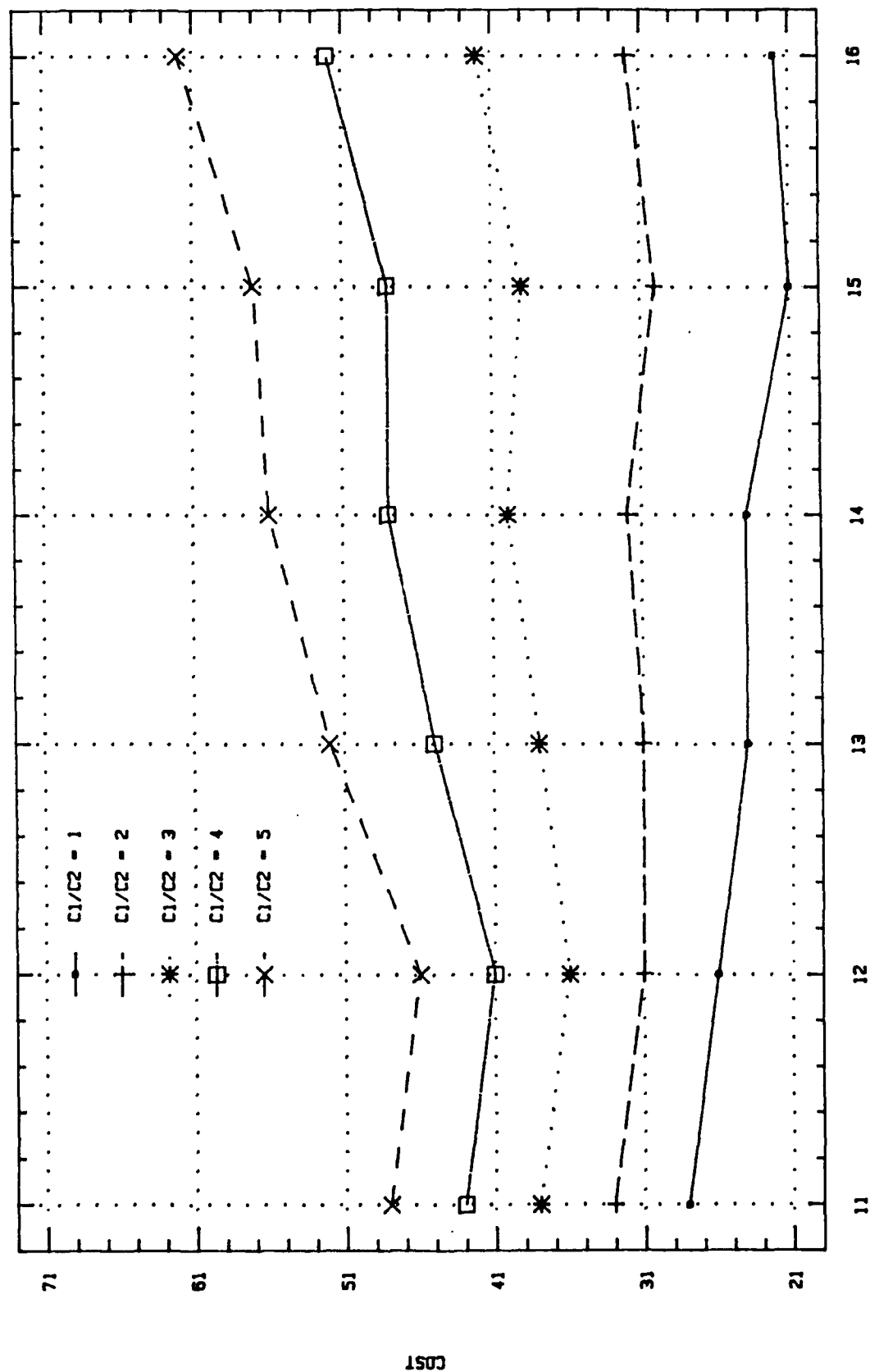


CRITICAL VALUE OF STATEMENTS

Figure 6

COST OF INCORRECT CLASSIFICATION

(STATEMENTS)



CRITICAL VALUE OF STATEMENTS

Figure 7

As was the case with factor analysis, it was found that using both C and S was no better than using C or S alone in the discriminant function. If a single variable is used, a very interesting and useful result is obtained. In this case, the coefficients in L become $b = 1$; then $L = C$, $L = S$. Using this result in \bar{L} , produces two cutoff values $\bar{L}_c = \bar{C}$ and $\bar{L}_s = \bar{S}$. Thus the mean values of $C = 2.53$ or 3 (same value as obtained with Chi-square) and $S = 14.29$ or 14 (value obtained with Chi-square = 13) could be used for C_c and S_s , respectively. The great advantage of this approach over the Chi-square technique is that C_c and S_s can be used directly, thus obviating the need for trial and error calculations with Chi-square.

CONCLUSION: C and S are valid with respect to the Discriminative Power criterion and either could be used to distinguish between acceptable ($C \leq 3$, $S \leq 13$) and unacceptable quality ($C > 3$, $S > 13$) for this and similar applications when this data can be collected. However, only one is needed (i.e., C is highly correlated with S and the correlation between E and C/S (normalized) is close to 0). It should be noted that it is less expensive to collect S than C.

Tracking

Ideally we want to track a metric against a quality factor over time for a **single component** (e.g., procedure). Unfortunately this type of data is not always readily available because a time history of corresponding quality factor and metric changes is required. This data was not available in this example. In lieu of this data, the Spearman Coefficient of Rank Correlation (r) can be used as a measure of the ordering of the metric in relation to the quality factor, with project being the 'component' (see below). Note, however, that (r) does not have a chronological ordering. Also, while ($r = 1$) implies perfect tracking, as defined previously, the converse is not true.

1. Compute the Spearman Coefficient of Rank Correlation (r) for E and C for Projects 1, 2, and 4 separately (Project 3 is not used because it has only one error). Correlation is lower for E and S than for E and C and is not shown. Compare (r) with $B = .7$ and α with .05. Procedures with errors are used rather than all procedures because the latter has too many ties in the sample. Rank correlation should not be used when there is a large number of ties. A moderate number of ties is tolerable [5,8].

Table 12

Spearman Rank Correlations (Error Procedures)

Project 1, N = 5 (small sample size)

	Complexity	Remarks
Errors	.8250	$r > .7$
α	.0990	$\alpha > .05$

Project 2, N = 12

Errors	.6723	$r < .7$
α	.0258	$\alpha < .05$

Project 4, N = 13

Errors	.2522	$r < .7$
α	.3824	$\alpha > .05$

RESULT: The desired result is $r > .7$ and $\alpha < .05$ (i.e. indication of on-zero correlation) for each project. Complexity does not track changes in Errors sufficiently for any of the projects. Therefore, Complexity is not valid with respect to Tracking. Also, Statements is not valid with respect to Tracking.

2. Subsequent to calculating (r), we were able to observe chronologically the procedures which comprise a project, so that for this example the project was the 'component' and the procedures that comprise the project were 'tracked'. A runs test was conducted for Projects 1 and 2 by assigning a '1' if M changed in the same direction as F (i.e. tracks) and a '0' if this was not the case (does not track). The runs test determines whether the binary sequences (runs) are systematic (i.e., M tracks F) or would be expected by chance.

RESULT: Projects 1 and 2 failed (did not track) the runs test.

Predictability

1. Make a scatter plot of E and C for procedures with errors to obtain a rough analysis of linearity [15].

RESULT: The dots on Figure 8 show the relationship.

Regression of Errors on Complexity

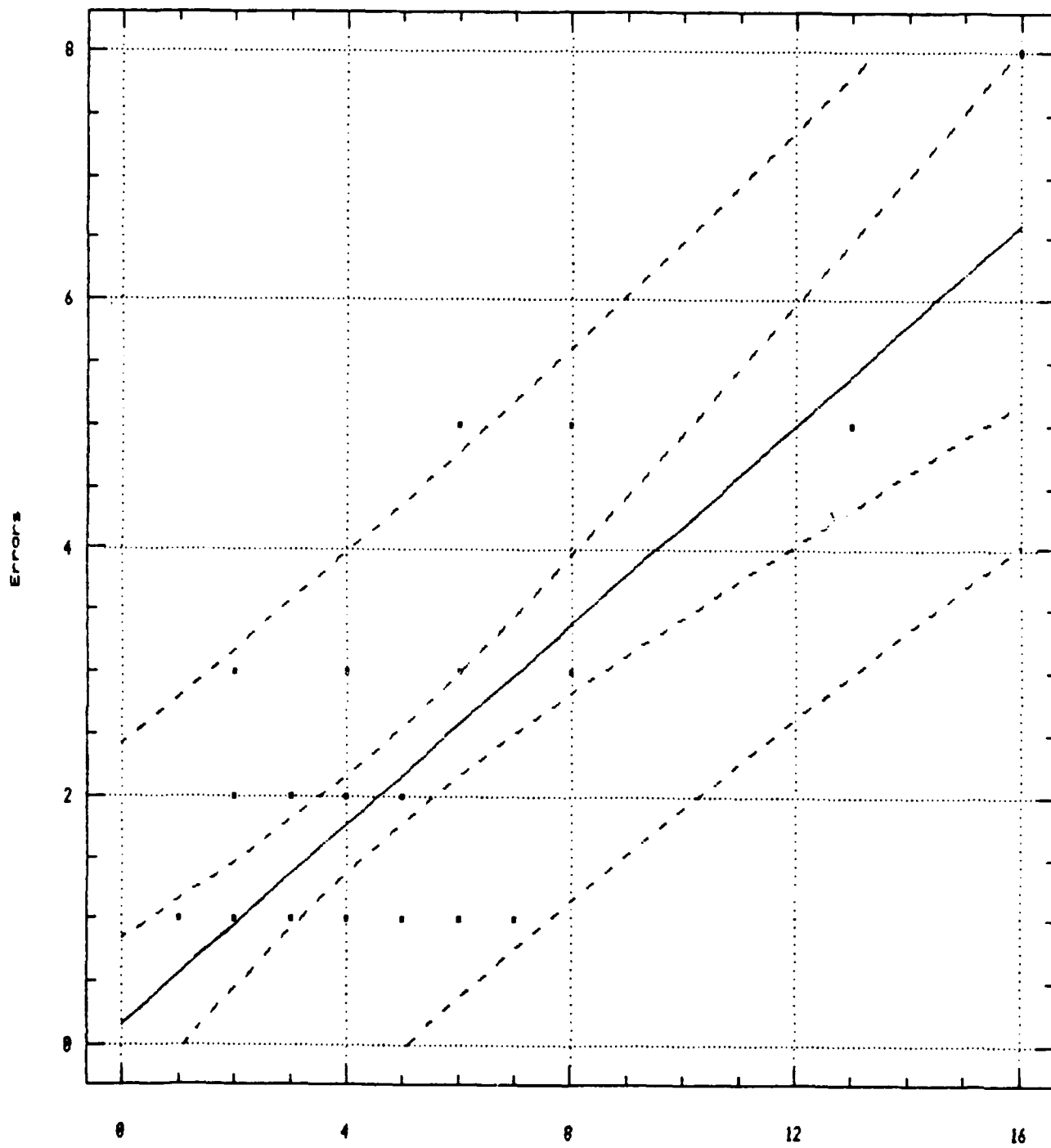


Figure 8 Complexity Metric

2. Perform a linear regression analysis of E on C for procedures with errors.

a. Test whether the assumptions of linear regression analysis hold for these data. Two of the important assumptions are: (1) E is normally distributed for given values of C and (2) the variances of E are equal for given values of C [15].

RESULT: For cases of C = 1 and 2, where there was an adequate sample size, tests were conducted and it was found that neither assumption holds. In addition, E was not normally distributed when all 112 procedures were used in the analysis. The best fit for E is a negative binomial distribution.

b. Examine the residuals of E (difference between observed and predicted as a function of C [15].

RESULT: Residuals increase with increasing C. This indicates that prediction error increases with increasing C. This is an undesirable result since we want prediction error to be independent of C.

c. The same results were obtained in a. and b. when all procedures were used.

3. Plot the regression model in Figure 8 for E on C for procedures with errors. The equation is: $E = .151 + .404C$. The inner band is the 95% confidence interval for average E (i.e., 95% chance that, for a given C, the estimate of average E will fall within the band) and the outer band is the 95% prediction interval of E (i.e., 95% chance that, for a given C, the estimate of E will fall within the band) [15]. The fit is worse for regression of E on S (not shown).

4. Compare Observed Errors with Predicted Errors (obtained from regression model) and note whether Predictability $< A = 20\%$, for $P \geq 80\%$ of the predictions.

RESULT: Table 13 indicates that Predictability $< 20\%$ ^{for} only 11 out of 31 cases, or 35%; the result is 16% when all procedures are used (not shown). Fails Predictability and Repeatability tests.

Table 13

Observed Errors	Predicted Errors	Prediction Error	Predica- bility (%)	Project
1	0.957831	0.04217	4.21686	1
5	2.572289	2.42771	48.55421	1
2	2.168674	-0.16868	8.43373	1
1	2.168674	-1.16867	116.86747	1
1	0.957831	0.04217	4.21686	1
1	0.554216	0.44578	44.57831	2
1	0.554216	0.44578	44.57831	2
1	0.554216	0.44578	44.57831	2
3	0.957831	2.04217	68.07228	2
3	3.379518	-0.37952	12.65060	2
1	1.765060	-0.76506	76.50602	2
3	2.572289	0.42771	14.25702	2
2	0.957831	1.04217	52.10843	2
1	1.765060	-0.76506	76.50602	2
2	2.168674	-0.16868	8.43373	2
1	1.765060	-0.76506	76.50602	2
8	6.608433	1.39157	17.39457	2
1	0.957831	0.04217	4.21686	3
1	2.572289	-1.57229	157.22891	4
1	2.168674	-1.16867	116.86747	4
5	3.379518	1.62048	32.40963	4
2	1.361445	0.63855	31.92771	4
1	1.361445	-0.36145	36.14457	4
1	2.975903	-1.97590	197.59036	4
1	2.168674	-1.16867	116.86747	4
3	1.765060	1.23494	41.16465	4
2	2.168674	-0.16868	8.43373	4
5	5.397590	-0.39759	7.95180	4
1	1.765060	-0.76506	76.50602	4
1	1.765060	-0.76506	76.50602	4
2	1.765060	0.23494	11.74698	4

5. Try non-linear single independent variable regression models.

RESULT: Several non-linear (eg., exponential) regressions of E on C for procedures with errors had lower correlation and worse fit (not shown) than the linear model.

6. Perform multiple linear regression analysis, using E as dependent variable and C and S as 'independent variables'.

a. Test whether the assumptions of the multiple regression model hold. An important assumption of this method is that the 'independent variables' are actually independent [15].

RESULT: The significant R between C and S of .833 for all procedures indicates dependence.

b. Examine the residuals of E for all procedures [15].

RESULT: Residuals increase with increasing C and S indicating that prediction error would increase with increasing C and S - an undesirable result.

c. Plot the multiple regression model and compare with results of Step 3 [15].

RESULT: The plots were made but are not shown because the fit is worse than in Step 3. For procedures with errors the regression equation is: $E = .174 + .437C - .00672S$. Statements contributes little to the relationship. The comparison between simple and multiple regression is summarized in Table 14, where F-Ratio is a measure of goodness of fit (generally, high value signifies good fit) and P is the percentage of predictions that are within the prediction error tolerance (A = 20%).

Table 14

	E vs. C Error Procedures	E vs. C All Procedures	E vs. C, S Error Procedures	E vs. C, S All Procedures
R	.783	.801	.785	.801
F-Ratio	46.1	196.9	22.5	97.6
P for A < 20%	35%	16%	35%	22%

CONCLUSION: Neither C nor S meets the Predictability criterion, either singly or in combination, for predicting E. Multiple regression has no advantage over single variable regression for these data. Also, the assumptions of both models are not satisfied. Therefore, both C and S are not valid with respect to Predictability.

Re-validate Metrics

Repeat all validation tests for C and S on future projects, keeping track of P, the Repeatability requirement (i.e., percentage of applications a metric must pass validity tests to be certified as valid).

Validate and Apply Metrics in Similar Environments

The final result of the validation exercise is that C and S are valid only with respect to the discriminative power criterion to support the quality control function. To the extent practical, apply C and S in applications and environments on future projects that are similar to this one.

SUMMARY AND FUTURE RESEARCH

We described a comprehensive metrics validation methodology that has six validation criteria, each of which supports certain quality functions. New criteria were defined and illustrated, including **consistency, discriminative power, tracking and repeatability**. It was shown that non-parametric statistical methods play an important role in evaluating whether metrics satisfy the validity criteria. A detailed example of the application of the methodology was presented. Although it was not an objective of our research, we found in the example that a single metric was sufficient to measure quality.

Future research is needed to extend and improve the methodology by finding answers to the following questions:

o To what extent are metrics that have been validated on one project, using our criteria, valid measures of quality on future projects -- both **similar** and **different** projects?

o Can optimum values of 'V', 'B', 'A', and 'P' be determined by balancing the 'cost' of setting the threshold of validity too high versus the 'cost' of setting it too low in order to reduce subjectivity in selecting these values?

o Can optimum critical values of metrics be found for the **discriminative power** criterion by using the 'costs' of misclassification in order to eliminate the calculation of these values by trial and error?

Acknowledgements

We wish to acknowledge the support provided to this research project by the Naval Surface Warfare Center, Dahlgren, Virginia. We also want to thank the members of the IEEE Standard for a Software Quality Metrics Methodology Working Group for many useful discussions and debates that helped inspire this work.

REFERENCES

1. A. A. Afifi and S. P. Azen, Statistical Analysis: A Computer Oriented Approach, Academic Press, 1972.
2. A. J. Albrecht and J. E. Gaffney, Jr., 'Software Function, Source Lines of Code, and Development Error Prediction: A Software Science Validation', IEEE Transactions on Software Engineering, Vol. SE-9, No. 6, November 1983, pp. 639-648.
3. Victor R. Basili, Richard. W. Selby, Jr., and Tsai-Yun Phillips, 'Metric Analysis and Data Validation Across Fortran Projects', IEEE Transactions on Software Engineering, Vol. SE-9, No. 6, November 1983, pp. 652-663.
4. Victor R. Basili, and David H. Hutchens, 'An Empirical Study of a Syntactic Complexity Family', IEEE Transactions on Software Engineering, Vol. SE-9, No. 6, November 1983, pp. 664-672.

5. W. J. Conover, Practical Nonparametric Statistics, John Wiley & Sons, Inc., 1971.
6. S. D. Conte, H. E. Dunsmore and V. Y. Shen, Software Engineering Metrics and Models, The Benjamin/Cummings Publishing Company, Inc., 1986.
7. Leonardo Felician and Graziella Zalateu, 'Validating Halstead's Theory for Pascal Programs', IEEE Transactions on Software Engineering, Vol. 15, No. 12, December 1989, pp. 1630-1632.
8. Jean Dickinson Gibbons, Nonparametric Statistical Inference, McGraw-Hill Book Company, 1971.
9. Harry A. Harmon, Modern Factor Analysis, The University of Chicago Press, Third Edition Revised, 1976.
10. Sallie Henry and Calvin Selig, 'Predicting Source-Code Complexity at the Design Stage', IEEE Software, Vol. 7, No. 2, March 1990, pp. 36-44.
11. IEEE Standard for a Software Quality Metrics Methodology (Draft), P-1061/D21, April 1, 1990.
12. IEEE Standard Glossary of Software Engineering Terminology, ANSI/IEEE Std 729-1983.
13. IEEE Glossary of Software Engineering Terminology (Draft), P729/610.12/D8, March 30, 1990.
14. T. M. Khoshgoftaar and J. C. Munson, 'Predicting Software Development Errors Using Software Complexity Metrics', IEEE Journal on Selected Areas in Communications, Vol. 8, No. 2, February 1990, pp. 253-261.
15. David G. Kleinbaum and Lawrence L. Kupper, Applied Regression Analysis and Other Multivariate Methods, Duxbury Press, 1978.
16. H. F. Li and W. K. Cheung, 'An Empirical Study of Software Metrics', IEEE Transactions on Software Engineering, Vol. SE-13, No. 6, June 1987, pp. 697-708.
17. John C. Munson and Taghi M. Khoshgoftaar, 'The Dimensionality of Program Complexity', Proceedings 11th International Conference on Software Engineering, May 1989, pp. 245-253.
18. Adam A. Porter and Richard W. Selby, 'Empirically Guided Software Development Using Metric-Based Classification Trees', IEEE Software, Vol. 7, No. 2, March 1990, pp. 46-54.
19. Elaine J. Weyuker, 'Evaluating Software Complexity Measures', IEEE Transactions on Software Engineering, Vol. 14, No. 9, September 1988, pp. 1357-1365.

APPENDIX A.

C: Complexity, S: Number of Source Statements (excluding comments)
E: Error Count

Procedures with No Errors

C	S	E	Project	C	S	E	Project
2	6	0	1	1	3	0	4
1	8	0	1	1	3	0	4
1	11	0	1	1	3	0	4
1	4	0	1	1	5	0	4
3	18	0	1	1	5	0	4
3	15	0	1	1	6	0	4
1	3	0	2	1	9	0	4
1	3	0	2	1	6	0	4
1	3	0	2	1	8	0	4
1	3	0	2	1	9	0	4
1	3	0	2	1	9	0	4
1	3	0	2	2	4	0	4
1	3	0	2	2	7	0	4
1	3	0	2	2	9	0	4
1	5	0	2	4	56	0	4
1	5	0	2	1	24	0	4
1	5	0	2	2	13	0	4
1	13	0	2	2	13	0	4
1	3	0	2	2	10	0	4
1	3	0	2	2	9	0	4
1	3	0	2	2	12	0	4
1	3	0	2	5	21	0	4
1	3	0	2	5	49	0	4
1	3	0	2	3	19	0	4
1	3	0	2	4	20	0	4
1	2	0	4	2	6	0	4
1	2	0	4	2	12	0	4
1	7	0	4	2	9	0	4
1	5	0	4	2	10	0	4
1	7	0	4	1	21	0	4
1	5	0	4	4	21	0	4
1	5	0	4	3	11	0	4
1	5	0	4	2	13	0	4
1	5	0	4	3	14	0	4
1	4	0	4	7	19	0	4
1	3	0	4	2	15	0	4
1	3	0	4	2	10	0	4
1	3	0	4	2	17	0	4
1	3	0	4	3	19	0	4
1	3	0	4	3	15	0	4
				2	15	0	4

Procedures with Errors

C	S	E	Project	C	S	E	Project
2	14	1	1	4	26	1	2
6	26	5	1	16	94	8	2
5	7	2	1	2	13	1	3
5	21	1	1	6	83	1	4
2	6	1	1	5	28	1	4
1	3	1	2	8	37	5	4
1	11	1	2	3	13	2	4
1	8	1	2	3	16	1	4
2	15	3	2	7	34	1	4
8	45	3	2	5	24	1	4
4	18	1	2	4	18	3	4
6	54	3	2	5	35	2	4
2	34	2	2	13	49	5	4
4	19	1	2	4	19	1	4
5	30	2	2	4	27	1	4
				4	17	2	4

DISTRIBUTION LIST

Strategic Systems Department Naval Surface Warfare Center Dahlgren, VA 22448 ATTN: Dr. William H. Farr	1
Mr. Roger Daugherty, Code 01B Navy Management Systems Support Office Naval Air Station Building R52-7 Norfolk, VA 23511-6694	1
Commander Space and Naval Warfare Systems Command Washington, D.C. 20363-5100 Attn: PMW 164-1, P. C. Davenport	1
Prof. Norman Schneidewind Code AS/Ss Naval Postgraduate School Monterey, CA 93943	25
Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
Knox Library Code 0142 Naval Postgraduate School Monterey, CA 93943	2
Computer Center Library Code 0141 Naval Postgraduate School Monterey, CA 93943	1
Administrative Sciences Department Library Code 54 Naval Postgraduate School Monterey, CA 93943	1
Research Administration Code 012 Naval Postgraduate School Monterey, CA 93943	1